MPI-IO Questionnaire
ver 0.3

The SIOF project has recently completed a working prototype of a high-level API for parallel I/O based off the emerging standard MPI-IO. Much of the extensive MPI-IO library has been implemented in the prototype -- we have learned a great deal about the issues and tradeoffs involved in the design of such a library. The prototype library is written using HPSS for lower-level media management. HPSS allows us to exploit certain advanced file operations such as 3rd party transfers.

Our next task will be to implement a production quality version of the MPI-IO library over HPSS which, when compared to the prototype, will offer improved performance and error handling. We are now reviewing the prototype and the validity of the various design tradeoffs we made. The following questions are intended to help the SIOF project make decisions about these tradeoffs, and to help us appraise the value of a number of new more elaborate optimizations now under consideration. Your cooperation, through answering the following questions, will provide very valuable information on how the library will be used and hopefully will enable us to make a better product

---

1.1) **Code Names:** What parallel or distributed codes are you now implementing/using? How about future codes?

1.2) **Programming Paradigm:** What paradigm do these codes use: shared-memory, distributed memory, or other? By ***shared memory***, I am referring to synchronization through barriers and locks such as using POSIX threads, LMPS or PARMACS on the Cray YMP, Cray J-90, DEC 8400, SGI Power Challenge, etc. By ***distributed memory*** , I am referring to synchronization via message passing primitives such as using MPI or PVM on the Meiko, or the Cray T3D, or multiple machines networked together. How about future codes (Shared memory on ASCI-Blue or ASCI-White, distributed memory on ASCI-Red or cluster of ASCI machines)?

1.3) **I/O Usage:** Do any of these codes perform a significant number of I/O operations? Do any of these codes write or read files of significant size? How about future codes?

1.4) **Write Operations:** Approximately how many files are generated/lengthened? Do the writes from a given node generally go to the same file or to multiple files? Is this write pattern static? Is the choice of the number of files based on the structure of the code, or an attempt to gain efficiency? If multiple files, would you prefer a single file if it could be done efficiently? How about future codes?

1.5) **Read Operations:** Approximately how may files are read? Do the reads from a given file generally go to the same node or to multiple nodes? Is the read-pattern static? Is the choice of the number of files based on the structure of the code, or an attempt to gain efficiency? If multiple files, would you prefer a single file if it could be done efficiently? How about future codes?

1.6) **Temporal Uniformity:** What determines when an I/O is to be issued (end of time-step, solution convergence, end of work sent from 'master', ...) ? Are the reads/writes from multiple nodes issued at approximately the same time, or do some nodes speed ahead and issue the read/write earlier? How about future codes?

1.7) **Write-Only Files:** How often are files opened for "write-only" access? In such cases, are the writes ever overlapping with differing data (note: we are not concerned with data overlaps due to multiple nodes sharing boundary conditions, ie. ghost cells, which are guaranteed to stay the same value from node to node throughout the life of the code)? What percentage of these writes are large writes (over 250KBytes)? How about future codes? How about future codes?

1.8) **Read-Only Files:** How often are files opened for "read-only" access? In such cases, are the reads ever overlapping with differing data (again, we are not concerned with data overlaps due to multiple nodes sharing boundary conditions, ie. ghost cells, which are guaranteed to stay the same value from node to node throughout the life of the code)? What percentage of these reads are large reads (over 250KBytes)? How about future codes?

1.9) **Small I/O Temporal Uniformity:** Do small I/O operations (reads/writes under 250 KBytes) occur near each other in time? Will they be directed to the same file, or to different files? Is the data fairly interleaved in a manner which could be combined to form one large read/write of the merged length? How about future codes?

1.10) **Part I Answer Confidence:** What are you basing your Part I answers on (e.g. general knowledge of the algorithm used, limited instrumentation/analysis of the code, detailed instrumentation/analysis of the code, ...)? How confident of your answers are you? How about future codes (will your involvement include being the principal code developer, on the code development team, having input into the design, ...)?